STUDIES IN
CLASSIFICATION, DATA ANALYSIS, AND KNOWLEDGE ORGANIZATION

H.-H. Bock · P. Ihm (Eds.)

Classification, Data Analysis, and Knowledge Organization

Models and Methods with Applications

Springer-Verlag

Hans-Hermann Bock · Peter Ihm (Eds.)

Classification, Data Analysis, and Knowledge Organization

Models and Methods with Applications

Proceedings of the 14th Annual Conference of the Gesellschaft für Klassifikation e.V., University of Marburg, March 12-14, 1990

With 148 Figures

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona

Part II: Knowledge organization, data bases, and information retrieval

Modelling, representation and organization of conceptual knowledge

Data bases, expert systems, information retrieval, and library systems

Terminology and classification

Small and Beautiful ?

Some remarks on evaluating microcomputer based library systems

S. Gradmann

Staats- und Universitätsbibliothek Hamburg,
Von-Melle-Park 3, D-2000 Hamburg

Abstract: The paper serves as an introduction for the presentation of different microcomputer-based library systems and suggests selected criteria for their evaluation by librarians working with rather small amounts of data and not beeing actively integrated in the structures of regional bibliographic networks or of complex local applications. The issue of different system environments for library use is beeing discussed in order to indicate the limits of this undertaking. The criteria developped are: Capacity of generating (1), concepts of data-maintenance, data-access and procedures providing the homogenity of data (2), functions related to cataloguing and subject indexing of bibliographic material (3), retrieval-oriented functions (4), aspects of ergonomy and documentation (5), functions and interfaces for data-exchange (6). The aspects of generating capacity and of functions related to data-exchange are given particular attention.

0. At first sight, the issue of this workshop seems clear and hardly ambiguous: it is defined as "micro-computer based programs for the administration of bibliographic data". Also, the organizing committee has cared for a limited audience; the invitation implies at least two common criteria of interest for participants:
- librarians working autonomously and not actively integrated in the structures of regional bibliographic networks or of complex local applications;
- librarians working with relatively small amounts of data compared again to regional or complex local area union catalogues.

I will try to meet these requirements in this introductive paper. However, let me first give some hints of warning as to the clearness of this issue, because it's definition does not provide a reliable guideline in the process of finding criteria for the evaluation of software.

As to the autonomy of an application: applications initially conceived as highly autonomous may lose much of this independence more quickly than it might have been intended. This loss of autonomy may be due to the structure of the specific institution beeing integrated in a larger administrative network, it may result from cooperation with partners wishing to exchange data, finally, even if it was initially intended to build up a bibliographic database autonomously, the availability of foreign data may force the user to work with these data: in any of these cases the user will have to give up more or less of this autonomy and mostly also want to do so, the advantages resulting from such a trend towards data-integration even between different hardware-systems and software-solutions seeming to prevail by far the restrictions which obviously resut from this process, too.

As to the dimensions of an application: there is a trend closely related to the point just discussed. Some applications will increase more quickly than originally planned, even if one does not suc-

cumb to a mere reflex of data-accumulation without practical need. This increase may have quantitative as well as qualitative aspects. The user might wish to integrate large amounts of outside data into his own information system - such a quantitative increase may still be supported by the original structure of the database. But if in the course of book processing it turns out to be necessary or at least favorable to add the aquisitions or lending system to the original cataloguing system, this might result in a qualitative increase which most certainly will exceed the limits of a simple basic system that seemed sufficient in the beginning.

The unforeseen occurrence of such functional or quantitative increases does not necessarily result from fals planning, because such developments sometimes are hard to preview. But once they impose themselves, such developments may render an initially satisfying software-solution an extremely heavy weight. In this case, the resulting complex solution will most probably be an extremely expensive and functionally inconsistent patchwork. Such a situation may in the end even lead to a complete change of the basic system - an expensive and mostly frustrating experience.

Therefore it is most important to consider the following questions (and to find clear and satisfying answers) before starting to work with whatever microcomputer-based system.
- First, one should ask, if the operating-system MS-DOS with all its internal limitations (also as far as communication with different computer-systems is concerned) may really be used as a basis for building up a library-system or if one should not consider alternative (for example UNIX-based) solutions right from the beginning.
- Even if MS-DOS is considered a sufficient basis the second question should be, whether the chosen software is is portable to such a degree as to offer reasonable conditions for a later change of the system environment.
- Thirdly, it has to be solved, whether the chosen software basis allows quantitative increase and functional additions as mentioned above within reasonable efforts.

Going back to the original subject, which of course cannot be treated thoroughly within this introductory paper, I will now develop some selected criteria , which I think of fundamental importance in the process of evaluating microcomputer-based user-systems for libraries. These criteria are:
1. Capacity of generating
2. Concepts of data-maintenance, data-access and procedures providing the homogenity of data
3. Functions related to cataloguing and subject indexing of bibliographic material
4. Retrieval-oriented functions
5. Aspects of ergonomy and documentation
6. Functions and interfaces for data-exchange.

1. Capacity of generating

Capacity of generating may be defined as the possibility to expand or modify a given software with regard to its basic data structures as well as to its user interface. A software would offer a highly developped generating capacity on a structural level, if the user is given the possibility for example to create new, logically distinct data areas, to add new fields to an existing record structure or remove these fields, to build logical links between fields or even beetween whole data areas or to attribute specific attributes or plausibility routines to fields or groups of fields. Capacity of generating the user interface could imply such features as free creation of forms for data entry and data display, modification of field tags or the flexible design of both structures and optical display of user menus.

It may become evident from the wide implications of such features, why this capacity of generating has been put in the first place here, for the availability of a well developed generating capacity may of course dispense with a whole number of detailled secondary questions. The concept of data-maintenance initially implemented by the vendor for instance will loose some of its importance, once such features of generating are provided, because this given structure may then be modified and adapted to the individual needs of one specific user. The capacity of generating the user interface may help to neglect ergonomically weak points of a software, for the design of entry and display areas as well as of user menus may be improved in this case.

The actual value of such features naturally depends on the comfort of access to the level of generating; the handling of these tools should not be too sophisticated and has to be clearly described. In most cases a library will decide to purchase a software product just because it is able to design an application system from the roots. The capacity of generating therefore would be of little additional use, if the handling of the generating tools confronts the user with problems comparable to those arising in the case of designing a library system in any current programming language.

Even if this generating capacity therefore seems to be fundamental, there are at least two serious reasons why the user should not be given universal liberties in this field. First, the user now becomes responsible to a high degree for the basic functions of his application; features of generating of course also imply the danger of making things worse by trying to improve them. The user has to pay for inconsistencies within the complex architecture of the whole software system caused by his own isolated generating efforts; the vendor is no longer responsible for such inconsistencies as long as the faults are not caused by disfunctions of the generating mechanism. Furthermore, such mechanisms of generating also have a security aspect: access to these tools must be controlled, a complete chaos would result from the unrestricted use of these features by several persons. Administration and use of the library system have to be separated on the level of staff organisation and this of course results in an increase of administrative and bureaucratic efforts.

More arguments for and against capacities of generation may be thought of, but the main factor in the decision process has to be the context of application. In any case, such features of generating provide a high security for a longlasting applicability of the basic software even in the case of modifications or functional expansions of the original application concept, once the necessary competence is provided by the context of their use. If such features are not availiable, the user would totally depend on the supplier of the basic system and could only hope, that the original design of the software comes up to his actual and future needs. He would depend on the system chosen for good and evil and might be forced to make compromises in an early stage and even more with the occurrence of new structural and functional demands. Plainly, he would have to adapt his needs to the frame forced upon his application by the features and restrictions of a static system. Therefore, a certain capacity of generating should be provided in any case within the reasonable limits indicated above mainly in order to avoid such frustrating compromises.

2. Concepts of data-maintenance and of data-access, procedures providing the homogenity of data

I will discuss the questions to be raised in this passage not so much under a technical point of view but rather with regard to the logical concept of data organisation. Even if the technical mode of its implementation prejudices its practical use to a high degree, one should make sure above all, that the system to be evaluated covers certain basic features as far as the logical structure of data and the mechanisms of access are concerned.

With regard to its concept of data organisation, the software should first use the available storage capacity economically, it should store bibliographic data with the least possible redundance, secondly it should provide effective means to ensure the homogenity of data, thirdly it should offer a mechanism of quick access to individual records.

2.1. The demand for economic administration directly implies certain technical features: as an example I only mention variable lengths of fields and records, which are of specific interest in the case of bibliographic data presenting enormous differences in the length of fields and records. But above all, certain structural demands for the concept of data maintenance derive from this aspect of storage economy, which correspond with the principles of minimum redundance and of data homogenity. 'Minimum redundance' is here used to describe the fact, that recurrent elements of title entries should ideally be stored in the database only once, or at least they should not have to be retyped and stored again separately on each occurrence.

To explain this argument which at the first glance semms to be somewhat trivial it is necessary to understand that bibliographic data may be classified into two general categories:
- on the one hand, there are generally variable elements, like for example the main title or the year of publication. Normally these are specific elements of a given title entry;
- on the other hand there are virtually recurrent elements which may turn up in several records like author, corporate source or serial titles but also place of publication, publisher and the part of the ISBN identifying the publisher.

Certainly it is not reasonable to register and store the elements of this second category again and again on each occurrence. A waste of storage capacity would be the result, the multiple storage of identical elements would render the database more and more redundant. Furthermore - and this already touches the second point mentioned above - working like this could lead to an unnecessarily heterogenous database mixed with uncontrolled double entries. Thus, identical informations could be registered with different spellings and under different authority forms, as a result, double entries would no longer be identified in any case, retrieval information would turn unreliable, because actually identical bibliographic features would be stored with different characteristics and could therefore no longer be put together by any retrieval strategy. Finally such a procedure would make little sense under ergonomical point of view: recurrent elements would have to be re-typed manually on each occurrence, changes concerning normalised elements would cause manual correction of each individual record concerned by this authority form (if one is at least still able to identify these records!) and data homogenity would again suffer seriously from this frustrating repetitive work. Furthermore, such recurrent elements often are part of identical complexes of bibliographic features: a publisher normally is joined to always the same place of publication and a specific part of the ISBN; a person is related to a number of references independently of the actual title entry, a serial title always is connected with a large number of constant elements such as corporate editor, place of publication, publisher etc. It certainly should not be necessary to register again and again all the elements which are part of such complex information once the constitutive element of this complex has been stored. This should applied to retrieval as well: complexes of information should generally be accessible by any single feature beeing part of such a complex.

Therefore, it seems necessary to keep such recurrent informations or complexes of information, which furthermore often have to be normalised in separate file areas. In that case a title entry is constituted by the fixed combination of variable elements in a title record with normalised elements in several authority files. These linked informations of a title entry kept in different data segments

of course have to be accessed and presented in retrieval and output processes as complete logic entities.

There are different technical modes of linking these different data segments which I will not discuss in this papaer by lack of time. Still, it is of fundamental importance that these different data areas are actually linked and that the software system does not simply copy authority informations into title entries. In this last case - only to mention one argument - modifications of authority informations would be of no consequence for the title entries virtually affected and the database would become inhomogenous.

2.2. This fundamental demand of data homogenity has to be furthermore supported by plausibility routines provided by the program, which in their turn form another condition for the identification of double entries. The definition of obligatory entry areas or the calculation of the last digit of the ISBN are examples for such mechanisms controlling the formal and syntactic correctness of data entry.

2.3. I will only mention the concept of data access at this point. It would certainly be insufficient and a waste of time if in order to adress a chosen field content the system had to go through the whole database sequentially in search of matching character strings. Therefore, a mechanism of quick access to individual records will have to be implemented; a classical example would be an indexing mechanism generating index informations from (possibly pre-manipulated) elements of records which in their turn provide direct access to these records.

In this context I wish to stress one point again: concepts of data maintenance, linking structures, plausibility routines, mechanisms for the identification of double entries and for the generating of index informations are only relatively useful as long as they are not accessible on the level of generating to be modified by the user.

3. Functions related to cataloguing and subject indexing of bibliographic material

3.1. In the field of formal description a clear maximum standard is given in the German speaking countries defined by the cataloguing rules "Regeln für die Alphabetische Katalogisierung" (RAK). A way how to apply fundamental demands of these rules in a database-oriented context has been proposed by MÜNNICH (1988) which seems to have been widely accepted. The conditions of implementing of a data structure according to these proposals derive directly from the system's structural capacity of generating, if such a structure is not already implemented, as this is the case with several systems offered in Germany.

Still, such a maximum standard of formal description may not be necessary for the needs of all libraries applications. Furthermore, the applicability of certain cataloguing rules in the context of database oriented systems has recently been doubted for good reasons (PAYER 1990). This is directly evident in the case of rules governing the order of entries in a card catalogue, which in an electronic context remain relevant only for sequential output of data. But even if one may choose different levels of conformity to these cataloguing rules, one principle should still be prevailing: the formal description has to allow clear identification of a bibliographic entity and it has to model bibliographic dependencies. This principle has been of fundamental importance in the process of working out the RAK, it maybe is of even more importance in the electronic processing of bibliographic data, for in this context minimal formal differences have further reaching consequences mainly for retrieval processes than it is the case in the traditional card catalogue.

Two fundamental demands directly result from this principle:
1. The program has to assure effective administration of normalised authority informations;
2. The program must offer modelling structures for complex hierarchical bibliographic dependencies.

3.2. The features implied by methods of suubject indexing certainly are less submitted to established standards as this is the case in cataloguing description. They depend to a much higher degree upon the needs and capacities of the individual user. I will therefore not go into the field of possible strategies of verbal or classificatory description. Still, independently from the strategy actually chosen by the user two demands mentioned above are supported, if methods of subject indexing are to be implemented:
1. Elements of subject indexing are recurrent informations which are to be normalised to a high degree. They have to be assigned to a clearly defined data area as with the other authority elements mentioned above and it must be possible to link them to complexes of variable bibliographic elements.
2. Complex hierarchical relations designed by the program must be comfortab and effective, if (poly-)hierarchic structures of classification or subject headings are to be used for searching.

4. Retrieval-oriented functions

The following point is difficult to treat generally and will therefore be touched only briefly. The necessity of a mechanism for quick access to individual records has already been mentioned. There is of course a wide range of other retrieval techniques of different relevance for the needs of a library. I recently met an extreme example: maybe for the first time - at least as far as library systems are concerned - a software vendor offered phonetically oriented search as a basic feature of his system.

The following considerations do not go that far; they exclusively concern techniques of string based free-text retrieval.

Generally, it should be possible to use all elements of records for retrieval globally or restricted to individual fields or groups of fields. It should be possible to use wildcards within retrieval terms and to truncate them at any position to the right or to the left. Retrieval terms should be possibly combined with field specific restrictions, it should be possible to express complex logic relations between individual terms of a complex search (this may be effected by the use of boolean operators, but it is possible to imagine other techniques).

Results should be presented by option in form of browsing lists sorted by a criterion freely chosen by the user.

Finally, the software should offer the possibility to hold retrieval results, to use them separately for further processing or for output (into a file or to a printing device) or to combine them with results of other retrieval steps.

5. Aspects of ergonomics and documentation

Ergonomical aspects have already been touched at several occasions, they too are difficult to treat systematically within the frame of this paper and therefore will not be discussed extensively here. Instead, I will only hardly scratch this issue in giving some examples of the problems involved.

Under ergonomic point of view, however, some errors should be avoided in any case although they do occur often. It is for example irritating and not helpful at all if function keys or sequences of keys are constantly assigned changing functions in different areas of a program.

Long, slightly structured and crowded screen sequences as for example in entry forms also may be a serious handicap for everyday work: if, in order to register just a handful of basic bibliographic elements the librarian has to find his way through a jungle of hundreds of data fields and if these fields are even distributed on four or five screens following hardly intelligible criteria this would certainly constraint the efficiency and motivation of the user.

Another constraining although well-intended error is the construction of unnecessarily scaled, complex and labyrinthic menu systems. The pull-down menus and other optical devices used may be as attractive as they want: the librarian's frustration might grow into annoying dimensions, if in order to enter an author not yet registered under his authority form he had first to store his actual title entry, move up two steps in the hierarchy of the menu system, move down again several steps to register the new authority information and then go back the whole way to link the title entry to the authority record.

These last two disadvantages might certainly be smoothed by an intelligent use of window techniques in the software design, which would allow the simultanous presentation of different data and function areas on the screen. Of course, an additional mistake in this case might be the excessive use of windows and colours which could possibly even increase the user's irritation by fripperies uselessly heaped on the screen.

I would consider the documentation of a program as part of its ergonomic aspects in a wider sense, because this documentation is a basic condition for the use of this system. Therefore, some words concerning this point might not seem superfluous here.

The following demands are rather trivial and obvious and still not always realised: the documentation should be exhaustive, correct, understandable, linguistically clear, easy to handle and it use should be facilitated by at least one index. Another feature is not that obvious: the documentation should not only explain correectly single steps and isolated functions, it also should offer general concepts for the intellectual integration of these isolated information elements to the user. As a necessary complement to informations of the type "if you want to obtain result Z, please, choose option A first, then press the buttons B and C and finish this step by choosing option D" the purpose of such key sequences should be made clear, their conceptual relation to other functional areas should be made evident and the documentation should show the common principles underlying the different parts of the complex system. All this can be done without recurring to the term 'philosophy' maybe used a little too often in such cases to state just general principles of software design.

Finally, the documentation should be integrated at least partly into the program in form of context oriented information screens, and the librarian should be given the possibility to modify such screens or to create new ones.

6. Functions and interfaces for data-exchange.

This is the last point of these considerations which I think to be of central and increasing interest. Today already (and even more in the near future) a librarian might use bibliographic online databases and want too integrate the downloaded files resulting from this work into his own system for further use. He might want to use large collections of bibliographic data availiable on CD-ROM instead of entering these data manually. He might be cooperating with partners also collecting bibliographic informations (still not necessarily using the same software) and who want to exchange data with him. One could imagine a large number of additional constellations, in which the integration of outside data or the output from the own system for further use by others become necessary features. The perspective of directly linking different database systems will not be discussed at this point, because even if this actually is becoming possible technically it still seems a little beyond the needs and (financial) means of the librarians and institutions I have in mind here.

Still, as a basic feature, the program should offer the possibility to integrate machine-readable bibliographic data availiable under various forms using mechanisms of automated input and it should be able to produce variable outputs not only on the screen or to a printer, but also in different file formats.

In both cases of data exchange one common problem is implied. A large amount of bibliographic data is currently availiable but only in highly differing file formats, each of which has its clearly defined structure but which, however, is not submitted to effective standards even within the national frame (this at least is the case in Germany). The complexity of these formats may vary from rather simple output files obtainable from most online databases containing just some basic bibliographic categories, one or two fields for subject indexing and an field containing the abstract to the highly sophisticated and analyticial hierarchical structures used for output from most of the bibliographic databases such as the "Maschinelles Austauschformat für Bibliotheken" (MAB 1), the various MARC-derivates currently used in Anglo-american countries or the just emerging standard UNIMARC. The problem arises from this babylonic plurality of file structures: in the case of data exchange two of these file structures have to be made congruent, a re-formating of data becomes necessary. Such re-formating calls for a clearly defined interface for data exchange, and in this sense one aspect of the complex field of data exchange may be reduced to the interfaces for data transformation provided by the program.

The program designer may choose between strategies in this field. He might of course implement a large number of predefined conversion tables for as many user formats as he just can get hold of. This seems to me a rather ineffective and laborious way of solving the problem, because despite all efforts one could never cover all formats virtually concerned and mainly because this strategy would make the user extremely dependent from the vendor in the case of any modification of format definitions: the designer would have to redefine the implemented interface according to these changes and only after that the librarian could use new data.

But the system's designer could also provide a generally defined interface for data exchange with integrated generating capabilities based for example on variable conversion tables and tools for data manipulation. This would enable the user to establish even complex correspondences between variable format definitions. In this case, of course, the user has to undergo the tedious and intellectually demanding task of analysing the format definitions and of defining the transformation rules by

himself. The producer therefore could add a number of predefined patterns for format conversion based on this general interface, which the librarian would have to modify only if necessary.

This second strategy seems to me the better one mainly because it provides the flexibility required in this field for the user by its generating capacity. Even if a library may be able to maintain fixed format conventions within its own application, it could hardly ever control the structures underlying the other component in the process of data exchange, the formats of input or of output files. Therefore a certain flexibility of the data interface seems indespensible. Furtermore, the argument may be applied analogously to the user's own application: even well developed generating tools and mechanisms of structural modification are only of little use as long as the interface for data exchange cannot be accorded to these modifications.

The procedures for data exchange as well as the modes generating the interface of course have to be intelligible and easy to handle as this has been stressed before with regard to the generating capacity in general. Automated input of data should require far less effort as the manual re-typing of these data, output should be much less demanding than the simple writing down of screen contents.

7. Conclusion

At the end I would like to recall some of my introductory remarks. Even if one has decided to work with an MS-DOS-based system one should care for good conditions of portability within this system. In this case one point for evaluation would be the programming language used for the implementation of the system: The commonly availiable language C certainly creates much better conditions under this point of view than an assembly language or any of the numerous BASIC-dialects. Also, the general architecture of a system may indicate conditions of portability to a multi-user and/or multi-process environment: the availiability of controlling mechanisms for access to files and records for example may prepare decisions in this sense.

I had to disregard many other important features of software systems for library use in this highly selective introduction. But even if I consider the points hitherto mentioned as crucial criteria I have to underline that I have mostly formulated maximum demands. Hardly any software could meet all these demands perfectly and there are conflicts between some of them I had no time to indicate here. It might therefore be a good strategy to consider all these criteria (and others I did not mention) without any restriction while looking for the perfect application software - but unfortunately such a perfect software does not exist.

The user therefore has to decide by himself which of these criteria are fundamental and which of them are less important with regard to his context of application.

Still, three of the complexes discussed in this paper, I think, have to be kept in mind in almost every context of evaluation:

1. High capacity of generating
2. Consistent and effective concepts of data administration
3. A generally defined interface for data exchange accessible on the generating level.

These three points seem to me an insurance for the survival of a library system in an almost protean technological context changing constantly and all too often just in unforeseen directions.

Bibliography/Further readings

BURTO, P.F. : Microcomputers in library and information systems. Aldershot 1986.

BURTON, P.F. : MS-DOS software for library and information applications. Aldershot 1990.

DYER, H., GUNSON, A. (EDS.) : A directory of library and information retrieval software for microcomputers. Aldershot 1986.

GRADMANN, S. : Katalogisierung mit dem PC. Mikrorechnergestützte Datenbanksysteme für die Verarbeitung bibliothekarischer Daten. Wolfenbüttel 1989.

LEHMANN, K.-D., STROHL-GOEBEL, H. (EDS.) : The application of microcomputers in information, documentation and libraries. Proc. 2nd International Conference on the Application of Microcomputers in Information, Documentation and Libraries. Amsterdam 1987.

MÜNNICH, M. : Katalogisieren auf dem PC - ein Pflichtenheft für die Formalkatalogisierung. Bibliotheksdienst 22 (1988) 841-856.

PAYER, M. : RAK für Online-Kataloge. Bibliotheksdienst 23 (1989) 1278-1288.

TEDD, L. : Computer-based library systems. A review of the last twenty-one years. Journal of Documentation 34 (1987), 2, 145-165.